A Beginner’s Course on
Reasoning About Imperative Programs

Kung-Kiu Lau

Department of Computer Science, University of Manchester
Manchester M13 9PL, United Kingdom
kung-kiu@cs.man.ac.uk

Abstract. Formal Methods teaching at undergraduate level has been going on at Manchester for a good number of years. We have introduced various courses based on different approaches. We have experienced the usual problems. To combat these problems, our approaches and our course contents have evolved accordingly over the years. In this paper we briefly trace this evolution, and describe the latest course, on reasoning about simple imperative programs, for first-year students who are half-way through our introductory programming course.

1 Introduction

Formal Methods teaching at undergraduate level has been going on at Manchester for a good number of years. We have run an introductory programming course which taught students how to read, and program from, VDM-like specifications. We have also had various courses on reasoning about programs which taught first-order logic, program specification and verification, and theorem proving techniques. On these courses, we have experienced the usual problems: the perception of Formal Methods as being too mathematical, too complex and completely impractical, coupled with an aversion to all things mathematical. To combat these problems, our approaches and our course contents have evolved accordingly over the years. In this paper we briefly trace this evolution, and describe the latest course, on reasoning about simple imperative programs, for first-year students who are half-way through our introductory programming course. We briefly trace its evolution, evaluate its success, and discuss the lessons learnt.

Apart from sharing our experience and the contents of our current course, we wish to convey the message that with suitable motivation and tool support, we believe it is possible to make the rudiments of Formal Methods accessible to beginner programmers. Moreover, it is highly desirable for such novices to be shown that testing alone is not sufficient for ensuring program correctness, and that Formal Methods are required for accomplishing the task properly.

2 Previous Courses

The current course has evolved from earlier courses on reasoning about programs. The latter courses started after a previous course on introductory programming using Pascal and VDM dropped out of the syllabus. In this section, we briefly describe all these courses.
2.1 Introductory Programming with Pascal and VDM

At first we had an introductory programming course for first-year students, where students were taught how to read specifications written in a notation similar to VDM [8], and how to write Pascal programs from these specifications. This was a compulsory course that ran for the full year. As an introductory programming course it had to teach beginner programmers the Pascal programming language from scratch. At the same time, the students were taught to read specifications written in a notation based on VDM. This notation was rather informal and not proper VDM at all. Even so, the students were not required to write any specifications, they only had to read specifications given to them. The course thus combined teaching Pascal programming with teaching the programming process of going from requirements, expressed as specifications, to program design and coding. To achieve this combination, special course material was developed that integrated the VDM-like notation into a programming process using Pascal. This material was published as a textbook [9] (shortly before the course was dropped from the syllabus! See below).

The course profile was as follows:

- **Lectures**: 72 lectures (3 a week)
- **Labs**: 10 lab sessions, 8 exercises
- **Tools**: None
- **Exam**: 3 hours, answer 4 questions

This course was dropped eventually when the syllabus for the whole first-year in our department was revised, replacing Pascal with SML as the first programming language. Although various versions of the course appeared subsequently as optional (half-year or one-semester) units in the third-year and second-year syllabi, eventually they disappeared altogether.

2.2 Reasoning About Programs I

In the new first-year syllabus, we introduced an optional half-year course on reasoning about programs. This consisted of 2 parts: (i) program verification; (ii) logic. Part (i) covered the specification and verification of a simple imperative language (with assignments, sequential composition, conditional commands, and while loops) and a simple functional language. For imperative programs, the semantics were based on predicate transformers: program specifications were expressed as assertions that represented pre- and post-conditions, and loop invariants were expressed by suitably defined assertions.

The emphasis was on proving correctness. For imperative programs, proof rules were given for the various constructs, and for functional programs, proof by induction was taught. There were no labs, however, so all verification exercises were done on paper only.

Part (ii) was not a continuation of Part (i). Instead it was a separate section that covered the basic concepts of first-order logic: formulae and their truth tables, normal forms, logical consequence, equivalence, validity, etc. The teaching was supported by labs based on Tarski’s World (see Section 4.1) for first-order logic.